DATA TRANSMISSION SYSTEM AND METHOD

[001] This is a Continuation of International Application PCT/DE00/02106, with an international filing date of June 28, 2000, which was published under PCT Article 21(2) in German, and the disclosure of which is incorporated into this application by reference.

FIELD OF THE INVENTION

[002] The invention relates generally to a system and method in the area of data transmission in an automated technical plant having distributed objects. In particular, the invention relates to a data transmission system method in which data is transmitted between a user program and a server program.

BACKGROUND OF THE INVENTION

[003] A control mechanism for support in a heterogeneously distributed programming environment is disclosed in David D. H. Lin, Behrooz Shirazi & Hassan Peyravi, *An Asynchronous Remote Procedure Call System for Heterogeneous Programming*, Proceedings of the Annual International Phoenix Conference on Computers and Communications, U.S. Los Alamitos, IEEE Comp. Soc. Press, Vol. Conf. 10, March 27, 1991 (1991-03-27), pp. 153-159, XP000299042 ISBN: 0-8186-2133-8 (hereinafter "Lin et al."). The Lin et al. mechanism is based on an asynchronous control algorithm for assigning a plurality of responses of a server to queries from a client.

[004] In Lin et al., a method is disclosed in which a time-stamped unique process ID is used to keep track of Remote Procedure Calls (RPCs) and responses thereto. A Remote Procedure Call is a protocol that one program can use to request a service from a program located in another computer in a network without having to understand network details. The RPC protocol uses the "client/server" model. The requesting program is a "client" and the service-providing program is the "server". Like a regular or local procedure call, a RPC is typically a synchronous operation requiring the requesting program to be suspended until the results of the remote procedure are returned. However, the use of lightweight processes or "threads" that share the same address space allows multiple RPCs to be performed concurrently.

[005] The unique process ID disclosed in Lin et al. is a concatenation of machine-ID, program-ID, dependency-ID and a local time-stamp. The unique process-IDs provide a mechanism to issue certain types of RPCs asynchronously, but process their replies in correct, i.e., synchronous, order.

[006] Additionally, in accordance with the Lin et al. method, three different data structures are used to implement the required control mechanism over the RPCs. The three data structures respectively deal with incoming calls, out-going calls and replies that need to be held for later processing.

[007] A client device, e.g., the device originating an RPC, processes calls and receives messages in an asynchronous fashion. To process RPCs, the client generates a header for each individual call and puts the header on the queue for out-going calls. The header contains the process-ID generated by the client and is used for claiming the reply and for ultimately keeping the desired order of the replies.

[008] When a client processes a received message, it first places the message on a buffer. The client then tries to match the header of the incoming call with the headers in the queue for out-going calls. If there are some headers with the same machine-ID, program-ID and dependency-ID but have a smaller timestamp than the timestamp of the incoming call, the client indicates that there are some RPC replies that have not yet returned and need to be processed before the incoming call can be processed. In this case, the reply is removed from the queue for out-going calls. The desired processing order of the replies is maintained by matching the IDs and comparing the timestamp of each header. The unprocessed replies in the buffer are checked every time a new message is received.

[009] In the server, messages are received, calls are dispatched and results are sent back to the client. The header for an incoming call is put on the queue dedicated for incoming calls. In the meantime, the server resets a timeout condition and prepares for the next incoming message. This monitoring operation continues until the timeout condition is reached. By the time the server stops monitoring the incoming messages on its communication port, control is switched over to the server dispatcher for dispatching the calls on the queue for incoming calls. As soon as the procedure is completed, the result is attached to the corresponding header in the queue for incoming calls and is sent back to the client. After the result is sent, the header of the call in the queue for incoming calls is removed from the queue.

[010] However, the Lin et al. system and method requires much overhead in the processing of RPCs. Thus, a method is desired that would simplify the solicitation and response of RPCs.

OBJECTS OF THE INVENTION

[011] One object of the present invention is to provide an asynchronous communication system between two processing devices over a network. In particular, in accordance with the invention, a simple but reliable method is proposed in which synchronization of applications in distributed systems is achieved by using an asynchronous communication channel.

SUMMARY OF THE INVENTION

[012] The invention relates to a system and a method for transmitting data between a local data processing system and a remote data processing system through an asynchronous transmission channel. Such a system is used, for instance, in the field of automation technology for operating and monitoring programmable controllers, such as stored-program controllers, numerical controls and/or drives.

[013] According to the invention, the objects mentioned above as well as others are achieved by providing a system for transmitting data between a local data processing system and a remote data processing system through an asynchronous transmission channel. The system utilizes a memory for storing at least one predefinable parameter provided for identifying a call of a first program of the local data processing device, such as a client or user program, sent to a second program of the remote data processing device, such as a server program. A predefinable parameter is also provided that is contained in a response of the remote data processing device to the local data processing device for identifying and/or synchronizing the response in the first program.

[014] The method serves to transmit data between a local data processing system and a remote data processing system through an asynchronous transmission channel. When a first program of the local data processing device, particularly a user program, sends a call to a second program of the remote data processing device, particularly a server program, a predefinable parameter provided for identifying a call is stored. In a response sent by the remote data processing device to the local data processing device, a second predefinable parameter for identifying and/or synchronizing the response of the first program is transmitted from the remote data processing device to the local data processing device, where it is evaluated.

[015] Data transmission from the local data processing system to the remote data processing system takes place through an asynchronous transmission channel. Therefore, upon receipt from the remote data processing system, the response data has to be matched and/or synchronized with the proper request. For this identification and/or synchronization of the response data, at least one predefinable parameter provided for identifying the call of a first program sent to a second program is stored in the local data processing device. The predefined parameter is integrated into the response data, which is returned by the remote data processing device to the local data processing device and is detected again in the local data processing device. This ensures simple identification and synchronization of the response data in the user program of the local data processing device.

[016] Reliable identification of the response(s) and thus assignment and synchronization of the programs can be achieved by providing the system with means for comparing the stored parameter and the predefinable parameter contained in the response.

[017] The predefined parameter can be implemented and configured with little complexity if the predefinable parameter is formed at least from parts of the IDL (Interface Definition Language) transmitted from the first program to the second program.

[018] One advantageous application in accordance with the present invention relates to the use of the system in the field of automation technology with distributed systems, for example, DCOM (Distributed Component Object Modeling) systems.

BRIEF DESCRIPTION OF THE DRAWINGS

[019] The invention will now be described in greater detail with reference to the drawing figures in which;

[020] FIG. **1** is a block diagram of a first exemplary embodiment of a system, in accordance with the present invention, for transmitting data between a user program and a server program, and

[021] FIG. **2** is a block diagram of a second exemplary embodiment of a system, in accordance with the present invention, for transmitting data between a user program and a server program.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[022] In accordance with the invention, FIG. 1 shows a block diagram of a first exemplary embodiment of a system for transmitting data between a user program 5 of a local data processing device 1 and a server program 6 of a remote data processing device 2. Communication between the local first data processing device 1 and the remote second data processing device 2 takes place through an asynchronous

transmission channel 3. Asynchronous in this context means that the respective transmission data from the first data processing device 1 and the reception data from the second data processing device 2 are transmitted in different transmission channels without synchronization. This is graphically indicated in FIG. 1 by arrow 3a, which represents a forward channel, and arrow 3b, which represents a reverse channel. The local first data processing device 1, by way of example, shows a user program 5, which includes the solicitation of information and/or data from a server program 6 of the remote second data processing device 2. To this end, the user program 5 of the local data processing device 1 sends a call 4 to the server program 6 of the remote data processing device 2. In the exemplary embodiment depicted in FIG. 1, a parameter 8 ("M1") is used as the predefined parameter. The M1 parameter is then integrated into the response 7 sent by the remote data processing device 2 through the reverse channel 3b to the local data processing device 1, where it is further processed by the user program 5. Based on the predefinable parameter 8 ("M1") the user program 5 has a means to match each response to the corresponding call and/or synchronize each call/response pair in the user program 5. Detection and identification of the predefinable parameter 8 is made possible by temporarily storing the predefinable parameter 8 in memory 9 of the first data processing device 1.

[023] Through the identification and synchronization of the user program 5 illustrated in FIG. 1, the user can reliably maintain a synchronous programming model, which also results in client applications that are easier to maintain. One advantageous application of the present embodiment is in regard to client applications as they relate to embedded systems, e.g., DCOM (Distributed Component Object Model) systems. DCOM is a set of Microsoft™ concepts and program interfaces in

which client program objects can request services from server program objects on other computers in a network. DCOM is based on the Component Object Model (COM), which provides a set of interfaces allowing clients and servers to communicate within the same computer.

[024] System performance particularly consists of storing the in-parameters from the stack and restoring them again prior to user callback. User callback is constructed to be identical to the original call. As a result, the client, or user, finds its in-parameters unchanged. Preferably, IDL (Interface Definition Language) is used to implement the predefinable parameter. IDL is a generic term for a language that lets a program or object written in one language communicate with another program written in an unknown language. In distributed object technology, it is important that new objects be able to be sent to any platform environment and discover how to run in that environment. An Object Request Broker (ORB) is an example of a program that might use an IDL to "broker" communication between two object programs.

[025] An IDL works by requiring that the interfaces of a particular program be described in a stub, or slight extension of the program that is compiled into the program. The stubs in each program are used by a broker program to allow them to communicate. As a result of using the IDL to implement the predefined parameter, a simple system with little complexity is achieved.

[026] FIG. 2 illustrates a second exemplary embodiment of a system in accordance with the present invention. The reference numerals in FIG. 2 are essentially the same as the reference numerals shown in FIG. 1. Thus, the description above, attendant to FIG. 1, is referenced in regard to the objects in FIG. 2 that have the same reference

numeral as objects in FIG. 1. However, unlike FIG. 1, in FIG. 2 a comparator 10 is provided. Comparator 10 compares the parameter data stored in memory 9 with the response data 7 transmitted by the server program 6. This comparison is used to ensure that the response 7 of the server 6 can be synchronously integrated into the user program 5 by means of the identification of parameter 8.

[027] In summary, the invention thus relates to a system and a method for transmitting data between a local data processing system 1 and a remote data processing system 2 through an asynchronous transmission channel 3. To ensure synchronization between the local data processing device 1 and the remote data processing device 2, it is proposed that when a first program 5 of the local data processing device 1, particularly a user program, sends a call 4 to a second program 6 of the remote data processing device 2, particularly a server program, at least one predefinable parameter 8 of the data to be transmitted is stored in the local data processing device 1. When the remote data processing device 2 sends a response 7 to the local data processing device 1, the predefinable parameter is provided for identifying and/or synchronizing the response 7 in the first program 5.

[028] The above description of the preferred embodiments has been given by way of example. From the disclosure given, those skilled in the art will not only understand the present invention and its attendant advantages, but will also find apparent various changes and modifications to the structures and methods disclosed. It is sought, therefore, to cover all such changes and modifications as fall within the spirit and scope of the invention, as defined by the appended claims, and equivalents thereof.